ITOS Plotting Users' Guide

Integrated Test & Operations System
5 October 2006

Introduction

The ITOS system provides the capability to do two-dimensional, x-y plots of real-time telemetry and of data stored in files. The system uses *plot definition files* to determine what is to be plotted and how, much like it uses page definition files to describe telemetry display pages: The plot definition files contain all the information needed to draw the desired plot, except the data.

Each plot definition consists of one or more charts. Each chart is a set of axes and the graphs drawn on them. Each graph is a plot of one STOL expression versus another. Of course, it can be more complex than that. The plot definition files offer a great many customizations, which the following chapters detail.

1 Plot Definition Files

Lets begin by defining some terms. A *plot* is a set of one or more charts in a certain layout. A *chart* is a set of axes (usually) and the graphs drawn on them. A *graph* is a visual representation of the relationship between two STOL expressions.

So, a plot definition file describes a plot window: what to plot and how to plot it. It contains a number of chart definitions, which each contain a number of graph definitions, and maybe axis definitions, too.

Here is an absolutely minimal plot definition:

```
plot minimal

chart
{
    graph
    {
        x gbl_gmtoff
        y psa3temp
    }
}
```

The above example defines a plot window (named `minimal`) that contains one unnamed chart that contains one graph of a power supply temperature versus ground time.

The file containing this definition should be named '`minimal.plot`', and live in one of the directories given in the mnemonic `GBL_PAGEPATH`. When you issue the STOL `PLOT` directive, it searches the list of directories for a file with the given name (forced to lowercase) with the '`.plot`' suffix. See section "PLOT" in *STOL Directives*.

Plot definitions are **not** line-oriented; they are free-format. The above example could just well have been given as:

```
plot minimal chart { graph { x gbl_gmtoff y psa3temp } }
```

The only rule is that STOL expressions for `x` and `y` that consist of multiple words must be enclosed in parentheses. (Otherwise, there's no way to tell when the expression ends and the next statement begins!)

The only exception to the free-format definition is comments: Comments begin with a pound sign (`#`) and continue to the end of the line.

Also, all plot keywords may be abbreviated down to the fewest characters which still are unique among the set of active keywords if all were abbreviated to the same number of characters. For example, all of the plot-level keywords may be abbreviated to a single character, since each still would be unique: `p` for `plot`, `b` for `background`, and so on. They also may be abbreviated to longer words, too, of course, such as `fore` for `foreground`. However, the axis keywords `min` and `max` could only be abbreviated down to `mi` and `ma` because `m` is ambiguous. Note that if you use an ambiguous abbreviation, one from the set of possible matches is chosen arbitrarily and a warning is issued.

Since plot definitions may be viewed hierarchically, we'll explore them that way, beginning with the statements that may appear at the top-level, outside of all chart definitions.

`plot`	First statement in a plot definition.
`title`	Specify the overall plot title.
`background`	Set the background color for all charts.
`foreground`	Specify the default foreground color.
`layout`	Specify the layout of charts.
`geometry`	Specify the plot window dimensions and placement.
`chart`	Define a chart.

Syntax Notation

In the syntax specifications given in the following sections, material enclosed in square brackets ([ ]) is optional. When any sequence of keywords or options are separated by a verticle bar (|), you must choose one from the sequence.

Statement keywords are typeset in a fixed-width font; for example, `keyword`. Statement parameters are set in italics; for example, *parameter*.

1.1 Plot statement

Syntax

`plot` *name*

Discussion

The `plot` statement must be the first statement in each plot definition file and is only allowed as the first statement in a plot definition file.

The `plot` statement names the plot, and it is expected to match the name of the plot definition file, except for the '`.plot`' extension. For the given example, the plot definition file name should be '`acswheelspeed.plot`'.

If the plot name and definition file name don't match, a warning is issued, but the plot still is run. The name given in the `PLOT` directive (that of the definition file) is used for the default title of the plot window on the display, and is reported in all event messages generated by the plot. (Basically, the `plot` statement isn't very useful . . . .)

1.2 Title Statement

Syntax

```
title string
```

Discussion

The `title` statment is used to specify an overall title for the plot. The *string* argument should be a quoted string.

1.3 Background statement

Syntax

```
background color
```

Discussion

The `background` statement specifies the background color to be used in all charts. The `background` statement is optional; the default background color is black.

The *color* argument is any color name understood by the X Window System (e.g. `red`, `black`, `"#112233"`, etc.). The actual list of valid color names is beyond the scope of this document; suffice it to say that most common color names are understood.

Because of considerations involved with printing plots on monochrome printers, we don't allow the individual charts in a plot to have different background colors.

1.4 Foreground statement

Syntax

```
foreground color
```

Discussion

The `foreground` statement is optional and is allowed at any level: plot, chart, and graph. A `foreground` at the plot level specifies the default foreground to be used in any subsequent charts that don't specify a foreground color. At the chart level, the statment gives the color of all axes and labels and the default color for graphs. At the graph level, the `foreground` statement gives the color in which the graph itself (line and points, for example) is to be drawn.

As for background, `color` is any color name understood by X.

1.5 Layout statement

Syntax

```
layout count columns | rows
```

Discussion

The layout statement gives you control over how the charts should be layed out in the plot window: in *count* columns or in *count* rows, where *count* is an integer. By default, charts are layed out in a single column (`layout 1 column`).

1.6 Geometry statement

Syntax

```
geometry spec
```

Discussion

The geometry statement gives yoy control over the dimensions and/or placement of the plot window. The *spec* argument is a standard X Window System geometry specification, which has the form *width*x*height*+-*xoff*+-*yoff*.

All values are in screen pixels. Offsets (*xoff* and *yoff*) may be preceded with a `+` *or* `-` sign. A positive x offset is measured from the left edge of the screen; a negative x offset is from the right edge. A positive y offset is measured from the top of the screen and a negative y offset is from the bottom.

You may give partial geometries to specify only the window dimensions or only the window placement. Here are some examples:

`geometry 600x300-0-0`
: Create the plot window 600 pixels wide and 300 tall in the bottom right hand corner of the screen.

`geometry 600x300`
: Create the plot window 600 pixels wide and 300 tall in a location chosen by the window manager.

`geometry -0+0`
: Create the plot window in the default dimensions (full height in roughly an 8.5 x 11 aspect ratio) in the upper right hand corner of the screen.

See documentation for the X Window System for more information.

1.7 Chart Statment

Syntax

```
chart [title] {
```

Discussion

The `chart` statement opens a chart definition, at least one of which is required in each plot definition file. A chart definition specifies a box with a set of axis in it on which a collection of graphs are drawn.

Between the `chart` keyword and the open brace, you may give a chart title. Titles containing blanks or special characters must be enclosed in quotes.

Within the chart definition, axes and graphs are defined. Here are the statements which may appear within a chart definition:

`foreground`
: See Section 1.4 [foreground stmnt], page 4.

`axis`
: Define a chart axis. You can set the axis min, max, and origin and control labels, titles, ticks, and grids.

`graph`
: Define a graph to draw on the chart. With this statement you define the x and y value pairs to plot along with line and point styles and colors.

End the chart definition with a closed brace (`}`) to match the open brace of the `chart` statement.

1.7.1 Axis Statement

Syntax

```
axis x|y {
axis 2nd_y Cx0 Cx1
```

Discussion

The `axis` statement opens an optional axis definition inside a chart definition. The axis definition controls how the axes are drawn and labeled and how they interact with the data being graphed on them.

After the `axis` keyword, give an `x` or `y` to identify the primary axis being defined, and then an open brace (`{`). Close the axis definition with the closed brace (`}`). Put between the braces one or more of the following statements to control the appearance and behavior of the axes:

`track`
: Move the axis min, max, and origin to keep the value plotted between min and max.

`grid`
: Display (or not) grid lines on the chart for this axis.

`origin`	Control axis origin placement.
`label`	Set the overall axis label.
`minimum`	Set the minimum axis value.
`maximum`	Set the maximum axis value.
`interval`	Set the interval at which value labels are placed on the axis.
`tick`	Set the number of tick marks between value labels.

The second y-axis is expressed as a function of the primary y-axis. The relationship is the first order polynomial $2nd\_y = Cx0 + Cx1 * y$, and the second y-axis is defined with the '`axis 2nd_y`' form given above, where *Cx1* is the slope and *Cx0* is the y-intercept.

By default, the axes are drawn so as to accomodate the data being graphed. For real-time plots, if a new data point would go outside the axes, they are expanded to hold the new point. The origin of both axes is chosen so that they are at minimum positive values or zero, if the data contains negative values. Only one y-axis is drawn by default.

1.7.1.1 Track Statement

Syntax

```
track on | off
```

Discussion

The `track` statement controls how the chart behaves when asked to plot a value outside of the given axis range (`minimum` & `maximum`). If tracking is on, and minimum and maximum axis values have been specified, the axis scale will track the values being plotted. The difference between the max and min values will be held constant, but the max and min values will be varied to keep the latest point in the view. Note that this axis shift may put older points outside the view.

Use this option on the x axis to set up stripcharts. Set `min` to zero and `max` to the desired time width of the window in seconds and turn tracking on.

If you don't give both a `max` and a `min` value for the axis, the `track` statement is ineffective. `track off` is the default, and so never need be specified, but it is provided for completeness.

1.7.1.2 Grid Statement

Syntax

```
grid on | off
```

Discussion

The `grid on` statement causes the chart to draw grid lines perpendicular to this axis, spaced to correspond to the axis labels. `grid off` is the default.

1.7.1.3 Label Statement

Syntax

```
label string
```

Discussion

The `label` statement in an axis definition sets the overall label to be drawn for the axis. If this is not specified, no axis title is drawn.

1.7.1.4 Origin Statement

Syntax

```
origin auto | zero | min | max
origin value
```

Discussion

In the first form above, the `origin` statement sets the rules to be followed for determining the origin for this axis. In the second form, the `origin` statement sets the axis origin to a fixed location subject to adjustment by axis tracking see Section 1.7.1.1 [track stmnt], page 7.

The automatic origin controls work as follows:

`auto`	This is the default setting and places the origin at the minimum positive data value. For data sets which contains negative values, the origin is set to zero.
`zero`	Place the origin at zero.
`min`	Place the origin at the minimum data value. This may be less than zero, and is used to keep the graphs from crossing the axis.
`max`	Place the origin at the maximum data value. This also keeps the graphs from crossing the axis.

If the origin is set to a specific value, and tracking is on, the origin is automatically shifted with the axis, maintaining its relative position to the axis min and max. For example: If `min` is set to 0, `max` is set to 10, and `origin` is set to 5, and the first point is at 20, the axis would be shifted to (about) a minimum of 10, maximum of 20, and origin set to 15.

1.7.1.5 Minimum Statement

Syntax

```
minimum value
```

Discussion

The `minimum` statement sets the axis minumum value. If this is not specified, the minimum value automatically is adjusted to accomodate all values graphed.

1.7.1.6 Maximum Statement

Syntax

`maximum` *value*

Discussion

The `maximum` statement sets the axis maximum value. If this is not specified, the maximum value automatically is adjusted to accomodate all values graphed.

1.7.1.7 Interval Statement

Syntax

`interval` *number*

Discussion

The `interval` statement attempts to set the interval at which axis labels are drawn.

1.7.1.8 Tick Statement

Syntax

`tick` *number*

Discussion

The `tick` statement attempts to set the number of tick increments drawn between each axis label. There will be one fewer marks drawn the the *number* given.

1.7.2 Graph Statement

Syntax

`graph [`*legend*`] {`

Discussion

The `graph` statement opens a graph definition, at least one of which is required in each chart definition. The graph definition identifies the x and y values to be plotted.

Charts containing more than one graph frequently use the same X values for each graph (for example, a chart might display several temperatures vs. time). The first graph in each chart must contain both `X` and `Y` statements; subsequent graphs inherit from the previous graph X or Y values not specified.

`foreground`	See Section 1.4 [foreground stmnt], page 4.
`point`	Set the graph point style. (dot, star, circle, . . .)
`line`	Set the graph line style. (solid, dotted, dashed, . . .)
`legend`	Set the graph's legend text.
`x | y`	Set the x or y value expression to be graphed.

End the graph definition with a closed brace (`}`) to match the open brace of the `graph` statement.

1.7.2.1 Point Statement

Syntax

`point` *style*

Discussion

The `point` statement is used to set the graph point style. The *style* argument may be one of:

```
none dot box triangle diamond star cross circle square table
```

The `dot` style is the default. Use `none` if you don't want point markers drawn.

1.7.2.2 Line Statement

Syntax

`line` *style*

Discussion

The `line` statement is used to set the type of line used to connect the data points on the chart. The *style* argument may be one of:

`none`	Don't draw lines connecting the points. Use this option to produce a scatter plot.
`solid`	Connect the points with a solid line.
`dot`	Connect the poinst with a dotted line.
`long-dash`	Connect the points with a dashed line composed of long dashes.
`short-dash`	Connect the points with a dashed line composed of short dashes.
`lsl-dash`	Connect the points with a dashed line composed alternating long and short dashes.
`dash-dot`	Connect the points with a line composed of alternating long dashes and dots.

1.7.2.3 Legend Statement

Syntax

legend [*text*] {

Discussion

Each chart contains a legend correlating each graph to a line on the chart. The `legend` statement provides one method of naming the graph for use in the legend. Normally, however, the graph name is given in the graph statement.

1.7.2.4 X or Y Value Statement

Syntax

x | y *expression*

Discussion

The `x` and `y` value statements give the STOL *expression* which the program is to evaluate to get the x or y value to be plotted for this graph. See section "Constants Variables and Expressions" in *ITOS STOL User's Manual* for more information on STOL expressions.

When plotting real-time data, the only symbols (that is, variables) permitted in an expression are telemetry mnemonics. When plotting data from files, the symbols used should match up with column heading from the data file.

In general, a graph definition must contain both an x and y value. For convenience, however, one (or both, but that's silly) may be omitted, and it will be taken from the previous graph defined for the same chart. Obviously then the first graph in each chart absolutely *must* contain both an x and a y value statement.

2 Plot Operation

The program dsp_plot is the ITOS plotting program. Each time you enter the STOL `PLOT` directive, a new instance of the plot program is started to instantiate the selected plot definition.

Several aspects of this program are of interest to the user:

2.1 Snap Menu

The plot program will display a pop-up menu when you move the mouse into it and press the right mouse button. This menu is for setting up and executing graphical snaps of the plot window. Choose the second selection, '`Snap properties...`', on the menu to set up the snap. The setup determines exactly how the first menu entry reads. Select the first menu entry to execute a snap.

2.2 Zoom Control

To zoom in in a chart, we draw a box around the area of the chart you want to be expanded to the whole chart. To draw the box, position the pointer on one corner of what will become the box, press the left mouse button, and drag the cursor to the opposite corner of the box. You'll see the box drawn as you drag the cursor.

When you release the mouse button, the chart will zoom the image so the box becomes the whole chart. You may repeat this procedure on the zoomed-in chart to zoom in further, if you want to.

To zoom back out to the original chart, type a lowercase '`r`' with the pointer on the chart.

2.3 Chart Properties

The commercial Motif widget which we use to create each chart is a product called XRT/graph from the KL Group. One feature of this product is that it encompases a pop-up properties manager that allows the user to change most features of the chart interactively.

To bring up the properties window, position the pointer over a chart and click the middle mouse button. You now should be able, using the properties window, to modify nearly everything about your chart, from axes and labels to the data values themselves.

Details on how to use the properties window is beyond the scope of this document.

2.4 Program Resources

Programs written for the X Window System generally employ *resources* to set various parameters needed by the program, and many of these are configurable by the user.

Here is a list of resources used by the ITOS plotting program, dsp_plot:

Name	**Type**	**Default**	**Explanation**
`interval`	integer milliseconds	n/a	Plot update interval.

`titleFont`	X font name		The font in which to set the plot title.
`dataSetSize`	integer samples	1000	The number of values stored for each graph.
`outputDir`	string	none	Name of the directory into which output files are written.
`graphFile`	string	none	Graphical output file name.
`defaultPrinter`	string	none	Default printer name.
`dataFile`	string	none	Data output file name.
`inputFile`	string	none	Data input file name.
`autoSnap`	boolean	false	If "true", turn on auto-snap.
`append`	boolean	false	If "true", append to data output file rather than overwriting it.

Note that `dataSetSize` probably should be settable in the plot definition file on a per-chart, or possibly a per-graph basis.

To modify any of the plotting program parameters shown above, you must create a file named .Xdefaults in your home directory. The entries in the file should be of the form:

```
<widget-name>*<parameter-name>: <value>
```

The plotting program's display is controlled by the widget named XDsp_plot. For example, to change the number of points stored for each graph to 2000, the 'dataSetSize' parameter of the XDsp_plot widget can be modified by adding the following line to your .Xdefaults file:

```
XDsp_plot*dataSetSize:  2000
```

Other resources also are used which control the Motif widgets used to build the plot window, including XRT/graph. Here is the widget tree for dsp_plot. Note that widget names are unique; where a given name appears more than once, they reference the same widget. There will be an explanatory notation enclosed in elipses whenever this occurs.

```
dsp_plot (XDsp_plot, a top-level shell)
          ↦ base (xmFormWidgetClass)
                    ↦ title (xmLabelWidgetClass)
                    ↦ form (xmFormWidgetClass)
                              ... continues as below ...
          ... or, if no plot title given ...
          ↦ form (xmFormWidgetClass)
                    ↦ chart (xtXrtGraphWidgetClass)
                    ... or, if layout diminsion > 1 ...
                    ↦ subform (xmFormWidgetClass)
                              ↦ chart (xtXrtGraphWidgetClass)
          ↦ pageMenu (xmRowColumnWidgetClass)
                    ↦ snapButton (xmPushButtonGadgetClass)
                    ↦ propsButton (xmPushButtonGadgetClass)
          ↦ snapForm (XmCreateTemplateDialog())
↦ form (xmFormWidgetClass)
          ↦ typeMenu (XmCreatePulldownMenu())
                    ↦ dsdmen_bw
                    (xmPushButtonGadgetClass)
                    ↦ dsdmen_color
                    (xmPushButtonGadgetClass)
                    ↦ dsdmen_bwfile
                    (xmPushButtonGadgetClass)
                    ↦ dsdmen_colfile
                    (xmPushButtonGadgetClass)
          ↦ typeOptionMenu (XmCreateOptionsMenu())
          ↦ destsel (xmRowColumnWidgtClass)
                    ↦ dsptxtcap (xmLabelGadgetClass)
                    ↦ dsdtxt (xmTextFieldWidgetClass)
```

Refer to Motif 2.0 and XRT/graph 3.0 documentation for resources specific to each widget in the tree.

3 Plot Data File Format

Non-numeric values in the data file are ignored, except that column headings are preserved.

Numeric values may be any legal C-language value form:

```
1, 1.2, 1e3, 1.2e-3, -4.2e+3, ...
```

Date values are of the form:

```
yy-ddd-hh:mm:ss.uuuuuu
```

where the '`.uuuuuu`' is optional, and any field may have fewer, but not more digits then called for above. Time values may be given in the folowing forms, where the '`.uuuuuu`' is always optional:

```
ddd-hh:mm:ss.uuuuuu
hh:mm:ss.uuuuuu
mm:ss.uuuuuu
```

The leftmost field in each case may contain one or more digits and all other fields may contain fewer, but not more, digits than called for above.

The string '`NV`' is recognized as a placeholder for "no value". This is what telemetry displays and sequential prints output for a value which has never arrived in the telemetry stream.

Values may be separated by spaces or commas, and the two methods may be intermixed. Commas exclusively are recommended because it allows the function to detect missing values. Two commas without an intervening value indicates a missing value, obviously.

An asterisk (*) preceding a value is interpreted as the "static flag" for that value. This presently has no meaning in time-on processing See section "TIMEON" in *STOL Directives*.

Text anywhere in the table is ignored. All values must be surrounded by whitespace, except that '`*`' may immediately precede any value. The number of columns in the table is set by the first row containing a value. Following rows will be filled with missing values or truncated as needed.